THE GTNPHONE DIALOG SYSTEM

David Stallard, Joshua Bers, Christopher Barclay

BBN Technologies
70 Fawcett St.
Cambridge, MA 02238, USA

ABSTRACT

BBN's GTNPhone is an over-the-telephone spoken language dialog interface to GTN, a DOD logistics information website. GTNPhone allows any user with a telephone, wired or cellular, to call in and retrieve information from the GTN web-site using voice only. No web browser or computer is required. The main body of the system is implemented in Java, with native code interfaces to commercial off-the-shelf components (such as a speech synthesizer) implemented in C. It runs on a Solaris workstation server located at BBN.

1. INTRODUCTION

BBN's GTNPhone is an over-the-telephone spoken language dialog interface to GTN [1], a DOD logistics information website. GTNPhone allows any user with a telephone, wired or cellular, to call in and retrieve information from the GTN web-site using voice only. No web browser or computer is required.

The GTN website provides users with the Netscape web browser access to large DOD databases that contain status information on US military cargo shipments and transport mission itineraries all over the world. Shipments are identified by a unique tracking code called a TCN, which can be used to access a large variety of information about the shipment, including its weight, volume, current status, its past and present location, what transport mission it is traveling on, etc. Part of an example GTN table is shown below:

TCN	Status	Loc	Time	Date
FB483372190022XXX	DEPART	KDOV Act	0210	7234
FB483372190085XXX	DEPART	KDOV Act	0000	7234

Unfortunately, a user without a computer and screen is not able to access this information, which presents problems for field personnel who are equipped only with a telephone or other voice-only communication device. The motivation for the GTNPhone system is thus to allow wider access to this crucial data, incorporating communication modalities such as the telephone which are more available, mobile, and convenient for personnel on the move.

An example session with GTNPhone is the following:

U: I want to check on TCN Charlie

S: Retrieving the cargo record from GTN… ok got it

S: TCN Charlie will arrive at Dover AFB at 1215 Zulu.

U: Where did it come from?

S: It is coming from Ramstein.

U: What time did it leave?

S: TCN Charlie left Ramstein at 0200 Zulu.

U: What plane was it on?

S: Tail number 50023.

A block diagram of the system is shown in Figure 1. At the center of the system is the dialog controller, which orchestrates the interaction of system components, using a network of dialog states, which provide it with rules for what to do and/or say in a given situation. The controller traverses this network in the course of a dialogue with the user. An earlier form of this architecture may be seen in [2]; other systems with some similarity to this design are [3] and [4].

All voice input and output goes through the telephone, which is connected to the serial port of the workstation. The speech synthesizer (Entropics TrueTalk), and the speech recognizer (BBN's Hark system), are also connected to the serial port. When a user calls the system, the dialog controller, in the initial WELCOME dialogue state, commands the synthesizer to play an initial prompt welcoming the user and asking him to say his name. The controller then places the speech recognizer in listening mode. When the user speaks, his speech goes through the recognizer and is transcribed into text. The text is passed through a language understanding component to produce a meaning frame. This meaning frame is then passed to the query formulator, which contacts the remote GTN website via HTTP to fetch the answer. The answer is formulated in terms of another meaning frame, which is passed to the Verbalizer component. The Verbalizer converts the answer meaning frame into an English text string, which is passed to the synthesizer and turned into synthesized speech that is played out to the user as a response. This completes the cycle, and the dialog controller continues on to the next dialogue state, until the user says "goodbye" or hangs up the phone.

In the rest of the paper, we go into more detail on each of these components, with special attention to how they work together in the complete operation of the system.

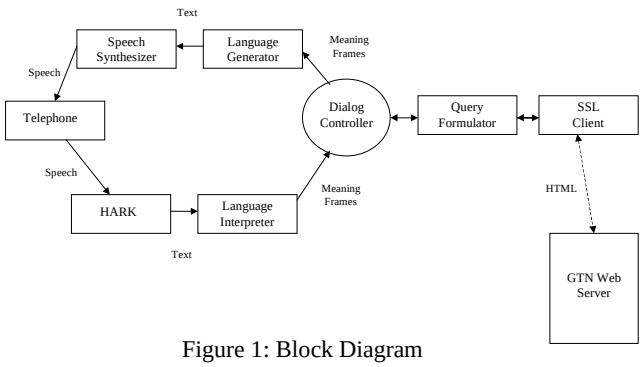

Figure 1: Block Diagram

2. CONTROLLING THE DIALOGUE

The complete network of dialog states in shown in Figure 2. The system starts off in the initial state, labeled WELCOME in the diagram, and asks the user for his name. If that is recognized as one of the names the system knows about, the system then proceeds to the VERIFIED state and from there to the PROMPT QUERY state, where it asks "How can I help you?". The user can either ask about a TCN previously known to both user and system ("What is the current status of TCN Charlie"), or he can introduce a new TCN. In the first case, the system goes to ANSWER QUERY, where it fetches the required data from GTN, converts the answer into an English sentence ("TCN Charlie arrived at Ramstein at 0800 Zulu yesterday") and speaks this to the user. In the second case, the user is led through a process of reading in a new TCN one group of characters at a time, which the system reads back to him to verify that it has heard them correctly. Once the new TCN has been successfully entered, the system gives the user a shorter tracking code (e.g. "Charlie", "Delta", etc.) which can refer to the TCN in subsequent sessions. The user can continue asking further questions about the various TCNs in his purview, and the system will continue traversing the state network accordingly until the user says "Goodbye" or hangs up the phone.

The concept of dialog state in this work is similar to that of [2], but is more detailed and grounded in object-oriented concepts. The different dialog states of the network correspond to the different types of situation that can arise in the course of the dialog. For example, in the PROMPT NAME state the system is prompting the user for his name and listening for his response, while in the ANSWER QUERY state it is retrieving the answer to his question from GTN and then speaking it to him. Dialog states specify rules for what to do or say in that state, based on what the user has said and/or what data the GTN application has returned. They may also have various parameters whose values are made use of by the rules, such as the meaning frame representations of user and system utterances, etc. We have found it natural to represent dialog states as classes in the Java language, where the rules are the methods of the class and the parameters are the instance variables.

There are two top-level subclasses of dialog state: prompt states and action states. Prompt states, such as PROMPT NAME, are those in which the system asks the user a question or otherwise prompts him to say something, and then listens for and interprets his answer, deciding what state to go to next based on it. Prompt states have the methods *constrainRecognizer*, *prompt*, *dontUnderstand*, and *action*.

Action states, such as ANSWER QUERY, are those in which the system performs some action and then decides what state to go to next. The system does not listen to the user at action states, but it may speak to him, as it does in ANSWER QUERY. Action states have only the method *action.*

The dialog controller maintains a pointer to the current dialog state, and invokes the methods of the current state in a fixed order. The last method invoked, *action*, returns a new dialog state which the controller then makes the current one, operating in a loop until the HANGUP state is reached.

As an example, consider the state PROMPT QUERY. A large variety of user responses are meaningful at this state, such as "When will it arrive?" or "I need to check on a new TCN", but many others are not, such as the words "yes" or "no", the user's name, etc. The state's *constrainRecognizer* method specifies which regions of the grammar are to be enabled, and which disabled.

The dialog controller then invokes the *prompt* method of the state, which at PROMPT QUERY returns the string "How can I help you?" (and in subsequent invocations, "What now?"). The controller takes care of passing this to the synthesizer, enabling recognition, and calling the language interpreter. If the utterance could not be recognized, or the language interpreter could not interpret it, the system says "Sorry, I couldn't understand that" and re-listens.

Otherwise, the dialog controller first checks to see if the meaning frame is one of the types that require special handling. If it is not, it calls the state's *action* method on the meaning frame. PROMPT QUERY's *action* method does not perform any side effect, but instead just specifies the next state to go to based on what the user said. If the meaning frame is of type NEW-TCN, the next state is TCN PROMPT. Otherwise, the next state is ANSWER QUERY.

If a meaning type requires special handling, it is dealt with by the dialog controller, and the state's action method is ignored. For example, at any point in the dialogue the user can say "Goodbye". The proper state to go to handle this is CONFIRM GOODBYE, but it would not be reasonable to expect the *action* methods of each and every state to include branching logic for this case. Instead, this meaning type is trapped in the dialog controller, which then transfers to the appropriate new state. Through this mechanism, the controller can create virtual links between states without their having to be explicitly specified. A spontaneous user hang-up is handled similarly.

The controller's special handling is useful in another way. GTNPhone allows the user to break out of a complex interaction with the system, such as reading in a nine-character TCN, by saying the word "abort". The meaning frame of type ABORT that results is intercepted by the controller, which then transfers to the state PROMPT QUERY. This means the user is never trapped in an interaction, and can always get back to a "home state".

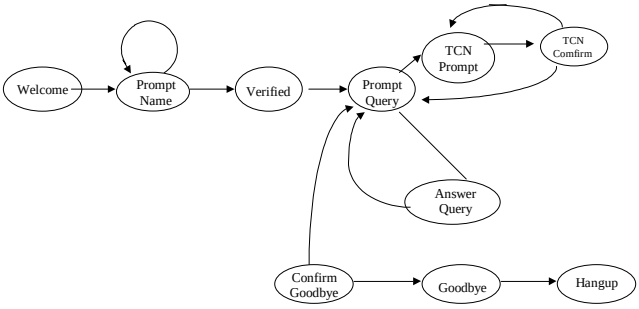

Figure 2: Dialog Network

3. SPOKEN LANGUAGE UNDERSTANDING

Spoken language understanding in GTNPhone is divided into three processing stages: speech recognition, sentence-level language understanding, and reference resolution.

Speech recognition is performed by BBN's Hark™ recognizer, running in a server described in [5]. The recognizer uses a finite state grammar embodying utterances in the GTN domain. The grammar rules are annotated with labels associated with syntactically and semantically distinct kinds of user utterances. Examples include a person answering yes or no, giving his name, asking a question, etc. Hark allows labeled grammar sections to be turned on or off, controlling which types of utterances can be recognized at a given time. GTNPhone uses this to increase recognition accuracy in certain situations, as when the user is responding to a system prompt (e.g. "Please say your name").

The individual words of the grammar ("Ramstein", "zero") are associated with tags that correspond to their referents in the GTN system ("ETAR", 0). The output of Hark is the complete set of these tags and labels (plus additional labels on grammar rules called regions). The sentence-level understanding component of the language interpreter module takes the regions and tags and produces from the recognition string a simple frame representing the meaning of the utterance. Frames consist of a type and zero or more role fillers. For example, the query:

USER: When will TCN Charlie arrive in Ramstein?

is represented as the meaning frame:

(ARRIVAL
datetime: ?
location: RAMSTEIN
equipment: CHARLIE)

The type of this frame is ARRIVAL, and it has fillers for the roles DATETIME, LOCATION and EQUIPMENT. Fillers may either be other frames, or terminal symbols denoting individual entities (e.g. RAMSTEIN). The distinguished filler symbol "?" indicates a question seeking the value of that role (here, the role DATETIME).

The Hark recognizer includes a rejection capability. If the user's utterance is judged to lie outside the grammar, or if the language interpreter cannot produce a meaning frame from Hark output, the language interpreter returns a frame of a the distinguished type DONT-UNDERSTAND.

The reference resolution sub-component is responsible for resolving implicit references in a user's query, by inferring them from context. As an example, consider the following context:

USER: Where is TCN Charlie coming from?

SYS: It is coming from Ramstein.

which the user may follow up with:

USER: When did it leave there?

or even:

USER: When did it leave?

In both cases, the user's complete meaning could be paraphrased "When did TCN Charlie leave Ramstein?". He makes use of pronouns ("it" or "there") or outright elision of elements (i.e. omitting the location in the second follow-up) to shorten what he has to say. As can be seen from this example, the antecedent of the implicit element can be found in either what the user himself said earlier ("it" = "TCN Charlie") or what the system said ("there" = "Ramstein").

GTNPhone tries to infer the proper fillers of pronouns and elisions by looking back at the history of the dialog. It uses a simple strategy of looking back up the predecessor chain of dialog states until it finds a referent of appropriate type. It then substitutes that referent as the role filler. The output of reference resolution is either a new meaning frame with all implicit elements resolved, or error if one could not be.

The reference resolution component performs one more useful service. When a user reads a new TCN into the system, the system assigns it a new tracking one word tracking name from the military alphabet ("alpha", "charlie", etc) and says it to the user. The name-table is saved by the system on disk, separately for each user. Thereafter, the user can refer to this new TCN by this code, without laboriously reading it in again, and the reference resolution component will match the name to the TCN, even in subsequent sessions with the system.

4. INTERFACING TO GTN

The Query Formulator takes a fully resolved query meaning frame and returns a new meaning frame that expresses the answer to that query, contacting the GTN website as necessary to retrieve needed information.

Retrieving data from the GTN website presents substantial technical challenges not present in ordinary interfaces to databases. There are multiple steps in the process. First, the system mimics the browser HTML-form interface by computing the correct CGI-encoded URL that will fetch the webpage for the TCN in question. A GET command with this URL is then passed to a Java HTTP client. The HTTP client then contacts the GTN server across an SSL (Secure Sockets Layer) connection, utilizing public key encryption. Once the page is fetched, it of course comes back as just an HTML string, in which the table of information for the TCN is encoded as a tab-formatted string embedded inside HTML tags. This is then parsed into a table data structure that implements basic JDBC functionality, from which the various data fields of the TCN can be accessed by attribute name. This table is cached in a hash array for the duration of the session, so that other data fields can be fetched from it as needed later in the session without accessing GTN again.

A new meaning frame is generated from the query, usually by substituting value obtained from GTN for the "?" symbols. It is possible, however, to produce a meaning frame of a completely different type instead, which is useful in some situations. For example, a STATUS query ("What is the status of TCN Charlie") can be answered by three different types of meaning frames, depending upon whether the item has arrived at, departed from, or is on hand at a particular base.

5. LANGUAGE GENERATION

Language is generated by the Verbalizer module, which takes an answer meaning frame as input and returns an English-language string as output. From the frame, the Verbalizer generates a clause whose main verb corresponds to the type of the frame and whose subject, object and other modifiers are generated from the role fillers of the frame. This process is driven by schemas associated with the frame's meaning type. These schemas specify what the main verb should be, as well as which roles are to be verbalized in which clause modifier positions.

For example, the meaning frame:

(ARRIVAL
date-time: 11/12/97-1400
location: ETAR
equipment: CHARLIE)

is handled by a schema specific to the ARRIVAL meaning type:

<equipment>
will arrive
at <location>
at <time>
on <date>

The tense of the clause is determined by comparing the date and time roles of the event to the current date and time. This governs what form the verb takes, e.g. "will arrive" vs. "arrived".

The individual role fillers themselves are verbalized according to separate schemas that are associated with the range type of the role. For example, locations, which are represented as codes, such as "ETAR", in the GTN data table, are verbalized as their full name form ("Ramstein Air Force Base"). These phrases plug in to the appropriate modifier positions in the clause generation schema.

The complete clause that would be produced by this example is "TCN Charlie will arrive at Ramstein Air Force Base at 1400 Zulu on November 12".

For the sake of concise and efficient communication, the Verbalizer makes use of indexical references where appropriate. For example, a date the same as the current date is verbalized as "today", whereas a date one day before the current one is verbalized as "yesterday", and so on. The system also uses days of the week (e.g. "Wednesday") if the date is within six days of the current date (the tense of the clause is sufficient for the hearer to discriminate between next Wednesday and last Wednesday).

6. SUMMARY AND STATUS

GTNPhone was demonstrated at the ALP (Advanced Logistics Program) briefings at the DARPA TIE facility in Washington DC on September 15, 1997. It was demonstrated again at various sessions from September 24 to October 1, 1997 at US Transcom. The last session was interactive and "hands-on," where people visited each station and could ask questions. GTNPhone performed well in both the fixed demo situation and in the hands-on sessions, and was very enthusiastically received.

In the near future, we plan to improve GTNPhone's usefulness by collecting data from potential users of the system. This will guide the development of the system in two areas: first, in determining what other types of information in GTN would be useful to access from a telephone dialog system, and second, what kind of language users are likely to employ in asking for that information.

We also plan to replace the current language understanding component, which is very simple and overly prescriptive, with a more powerful statistical one for broader coverage. Additional areas of future work include improved strategies for error recovery and the improved use of discourse context in language generation.

7. ACKNOWLEDGEMENTS

The work reported here was supported by the Defense Advanced Research Projects Agency and was monitored by the Office of Naval Research under Contract No. DABT63-94-C-0061. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the United States Government.

8. REFERENCES

[1] USTRANSCOM Global Transportation Network website. https://www.gtn.transcom.mil/

[2] Stallard, D. "The Initial Implementation of the BBN ATIS4 Dialog System". *Proceedings of the Spoken Language Systems Technology Workshop*, January 1995. Austin, TX. ARPA/SISTO

[3] Bratt, H., Dowding, D., and Hunicke-Smith, K. "The SRI Telephone-based ATIS System". *Proceedings of the Spoken Language Systems Technology Workshop*, January 1995. Austin, TX. ARPA/SISTO

[4] Seneff, S., Zue, V., Polifroni, J., Pao, C., Hetherington, L., Goddeau, D., and Glass, J. "The Preliminary Development of a Displayless PEGASUS System". *Proceedings of the Spoken Language Systems Technology Workshop*, January 1995. Austin, TX. ARPA/SISTO

[5] Bers, J., Miller, S., Makhoul J. *"Designing Conversational Interfaces for Mobile Networked Computing"*. In Proceedings of the Workshop on Perceptual User Interfaces. Banff, Alberta, Canada Oct 19-21, 1997.